

|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное бюджетное образовательное учреждение высшего образования  **«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Лабораторная работа № 4**

по дисциплине

«Структуры и алгоритмы обработки данных»

на тему:

**«**Задача о заполнении рюкзака**»**

**Выполнил:**

Учебная группа: 1бПМ   
ФИО: Борисов И. М.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель лабораторной работы:**

Должность: старший преподаватель

Звание: б/з

ФИО: Кутейников И. А.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Цель:

В задаче о рюкзаке есть набор предметов. Каждый предмет имеет

название, вес и ценность. Требуется сложить вещи с максимальной

стоимостью в рюкзак, имеющий ограничение по весу.

Реализовать программу решающую задачу о заполнении рюкзака с

помощью:

1 Рекурсивного метода;

2 Метода динамического программирования;

3 Жадного алгоритма.

Для жадного алгоритма реализовать стратегии:

−

заполняем в первую очередь предметами с максимальным весом

(нечетные варианты)

−

заполняем в первую очередь предметами с максимальным

соотношением цена/вес (четные варианты)

В программе должен присутствовать класс «Предмет», обладающий

полями: название, вес, цена; и класс «Рюкзак», обладающим полями:

максимальный вес, текущий вес, список предметов в рюкзаке.

Реализовать меню:

1 Заполнение списка предметов из файла

2 Добавление предмета

3 Изменение предмета

4 Удаление предмета

5 Задание максимального веса рюкзака

6 Просмотр содержимого рюкзака

7 Выбор способа решения задачи

8 Сравнение способов решения

Код:

class Ui\_Form(object):

    def setupUi(self, Form):

        Form.setObjectName("Form")

        Form.resize(728, 692)

        self.ChoosingItems\_table = QtWidgets.QTableWidget(Form)

        self.ChoosingItems\_table.setGeometry(QtCore.QRect(20, 20, 511, 651))

        self.ChoosingItems\_table.setAccessibleName("")

        self.ChoosingItems\_table.setRowCount(0)

        self.ChoosingItems\_table.setColumnCount(4)

        self.ChoosingItems\_table.setObjectName("ChoosingItems\_table")

        item = QtWidgets.QTableWidgetItem()

        self.ChoosingItems\_table.setHorizontalHeaderItem(0, item)

        item = QtWidgets.QTableWidgetItem()

        self.ChoosingItems\_table.setHorizontalHeaderItem(1, item)

        item = QtWidgets.QTableWidgetItem()

        self.ChoosingItems\_table.setHorizontalHeaderItem(2, item)

        item = QtWidgets.QTableWidgetItem()

        self.ChoosingItems\_table.setHorizontalHeaderItem(3, item)

        self.DeleteButton = QtWidgets.QPushButton(Form)

        self.DeleteButton.setGeometry(QtCore.QRect(580, 150, 121, 28))

        self.DeleteButton.setObjectName("DeleteButton")

        self.ChangeButton = QtWidgets.QPushButton(Form)

        self.ChangeButton.setGeometry(QtCore.QRect(580, 210, 121, 28))

        self.ChangeButton.setObjectName("ChangeButton")

        self.PutInBpButton = QtWidgets.QPushButton(Form)

        self.PutInBpButton.setGeometry(QtCore.QRect(580, 270, 121, 28))

        self.PutInBpButton.setObjectName("PutInBpButton")

        self.BackButton = QtWidgets.QPushButton(Form)

        self.BackButton.setGeometry(QtCore.QRect(590, 640, 93, 28))

        self.BackButton.setObjectName("Back")

        self.retranslateUi(Form)

        QtCore.QMetaObject.connectSlotsByName(Form)

        self.loadDataFromFile()

    def retranslateUi(self, Form):

        \_translate = QtCore.QCoreApplication.translate

        Form.setWindowTitle(\_translate("Form", "Form"))

        item = self.ChoosingItems\_table.horizontalHeaderItem(0)

        item.setText(\_translate("Form", "Name"))

        item = self.ChoosingItems\_table.horizontalHeaderItem(1)

        item.setText(\_translate("Form", "Weight"))

        item = self.ChoosingItems\_table.horizontalHeaderItem(2)

        item.setText(\_translate("Form", "Cost"))

        item = self.ChoosingItems\_table.horizontalHeaderItem(3)

        item.setText(\_translate("Form", "Choose"))

        self.DeleteButton.setText(\_translate("Form", "Delete"))

        self.ChangeButton.setText(\_translate("Form", "Change"))

        self.PutInBpButton.setText(\_translate("Form", "Put into backpack"))

        self.BackButton.setText(\_translate("Form", "BackButton"))

    def save\_table\_data\_to\_csv(self):

    # Открываем файл для записи

        with open(f"C:\\Users\\Mrgor\\Desktop\\Programs\\For\_study\\Python\_labs\\First\_year\\Lab8\\items", 'w', encoding='UTF-8', newline='') as file:

            writer = csv.writer(file, delimiter='\t')

            # Получаем заголовки столбцов таблицы

            headers = [self.ChoosingItems\_table.horizontalHeaderItem(i).text() for i in range(self.ChoosingItems\_table.columnCount() - 1)]

            # Записываем заголовки в первую строку CSV файла

            writer.writerow(headers)

            # Записываем данные таблицы

            for row in range(self.ChoosingItems\_table.rowCount()):

                row\_data = [self.ChoosingItems\_table.item(row, col).text() if self.ChoosingItems\_table.item(row, col) else '' for col in range(self.ChoosingItems\_table.columnCount())]

                writer.writerow(row\_data)

    def close\_window(self):

        self.table = Table\_Ui.MyDialog()

        self.table.open()

        self.close()

    def read\_data\_from\_file(self):

        try:

            with open(f"C:\\Users\\Mrgor\\Desktop\\Programs\\For\_study\\Python\_labs\\First\_year\\Lab8\\items", 'r', encoding='UTF-8') as file:

                lines = file.readlines()[1:]  # Пропускаем заголовок

                data = [line.strip().split() for line in lines]

            return data

        except FileNotFoundError:

            print("Файл не найден.")

            return []

    def write\_data\_to\_file(self, data):

        try:

            with open(f"C:\\Users\\Mrgor\\Desktop\\Programs\\For\_study\\Python\_labs\\First\_year\\Lab8\\items", 'w', encoding='UTF-8') as file:

                file.write("Name\tWeight\tCoast\n")  # Заголовок

                for line in data:

                    file.write("\t".join(line) + "\n")

        except Exception as e:

            print(f"Ошибка при записи в файл: {e}")

    def loadDataFromFile(self):

        data = self.read\_data\_from\_file()

        self.ChoosingItems\_table.setRowCount(len(data))

        self.ChoosingItems\_table.setColumnCount(4)  # Устанавливаем количество колонок

        for row\_num, row\_data in enumerate(data):

            for col\_num, item in enumerate(row\_data):

                self.ChoosingItems\_table.setItem(row\_num, col\_num, QTableWidgetItem(item))

            # Добавляем флажок в четвертую колонку

            checkbox\_item = QTableWidgetItem()

            checkbox\_item.setFlags(checkbox\_item.flags() | Qt.ItemIsUserCheckable)

            checkbox\_item.setCheckState(Qt.Unchecked)

            self.ChoosingItems\_table.setItem(row\_num, 3, checkbox\_item)

    def delete\_selected(self):

        data = self.read\_data\_from\_file()

        rows\_to\_delete = []

        for row in range(self.ChoosingItems\_table.rowCount() - 1, -1, -1):

            item = self.ChoosingItems\_table.item(row, 3)

            if item.checkState() == Qt.Checked:

                rows\_to\_delete.append(row)

        for row in rows\_to\_delete:

            self.ChoosingItems\_table.removeRow(row)

            del data[row]

        self.write\_data\_to\_file(data)

    def save\_checked\_rows\_to\_file(self):

        # Открываем файл для записи

        with open(f"C:\\Users\\Mrgor\\Desktop\\Programs\\For\_study\\Python\_labs\\First\_year\\Lab8\\Backpack", 'w', newline='', encoding='utf-8') as file:

            file.write("Name\tWeight\tCoast\n")

            writer = csv.writer(file, delimiter='\t')

            # Проходимся по строкам таблицы

            for row in range(self.ChoosingItems\_table.rowCount()):

                # Проверяем, установлен ли чекбокс в четвертой колонке

                checkbox\_item = self.ChoosingItems\_table.item(row, 3)

                if checkbox\_item.checkState() == Qt.Checked:

                    # Получаем данные из первых трех колонок

                    row\_data = [

                        self.ChoosingItems\_table.item(row, col).text() if self.ChoosingItems\_table.item(row, col) else ''

                        for col in range(3)

                    ]

                    # Записываем данные в файл

                    writer.writerow(row\_data)

class MyDialog(QDialog, Ui\_Form):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.setupUi(self)

        self.BackButton.clicked.connect(self.close\_window)

        self.DeleteButton.clicked.connect(self.delete\_selected)

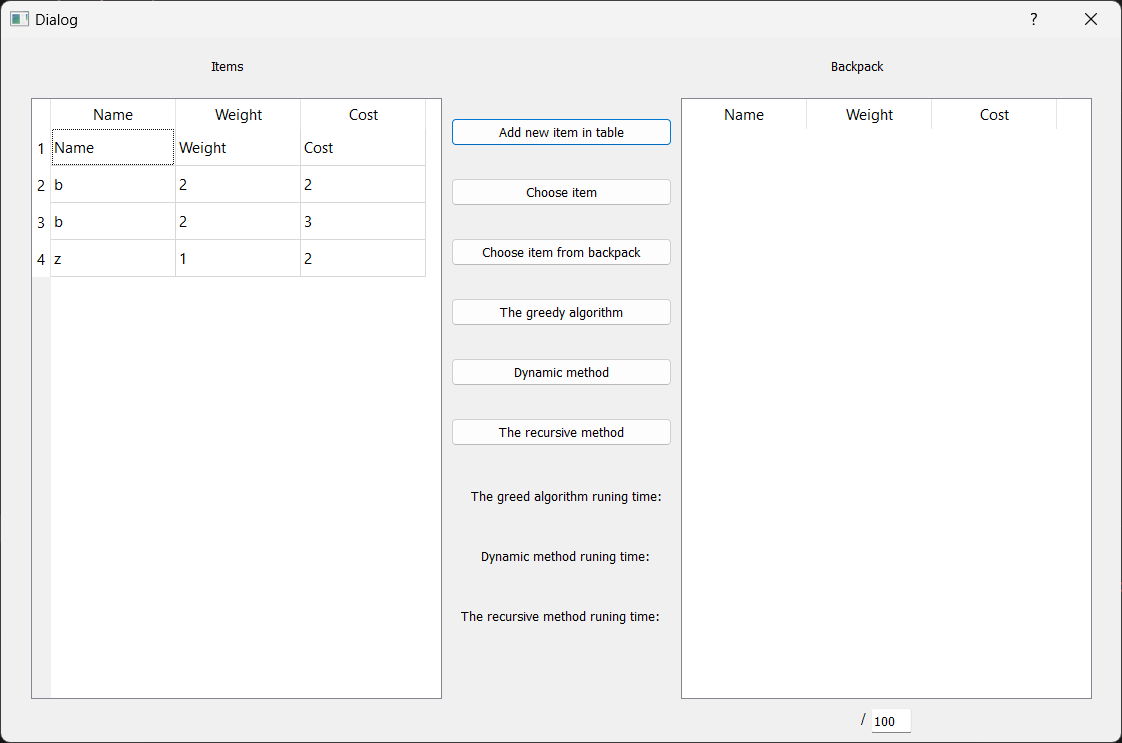
        self.ChangeButton.clicked.connect(self.save\_table\_data\_to\_csv)

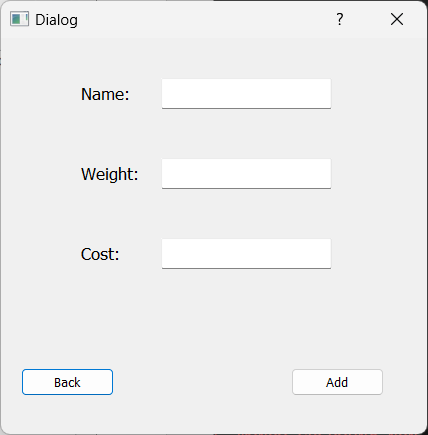
        self.PutInBpButton.clicked.connect(self.save\_checked\_rows\_to\_file)

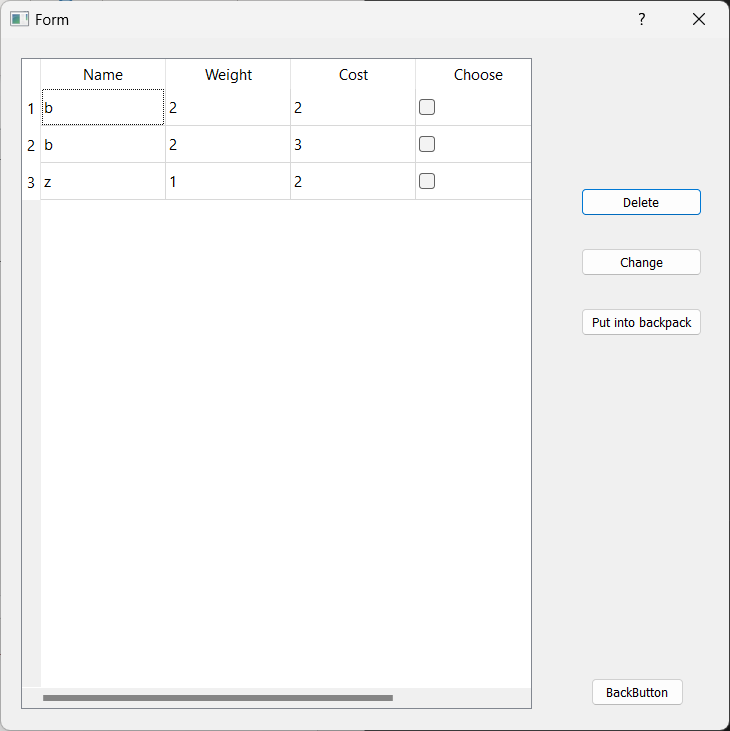
…

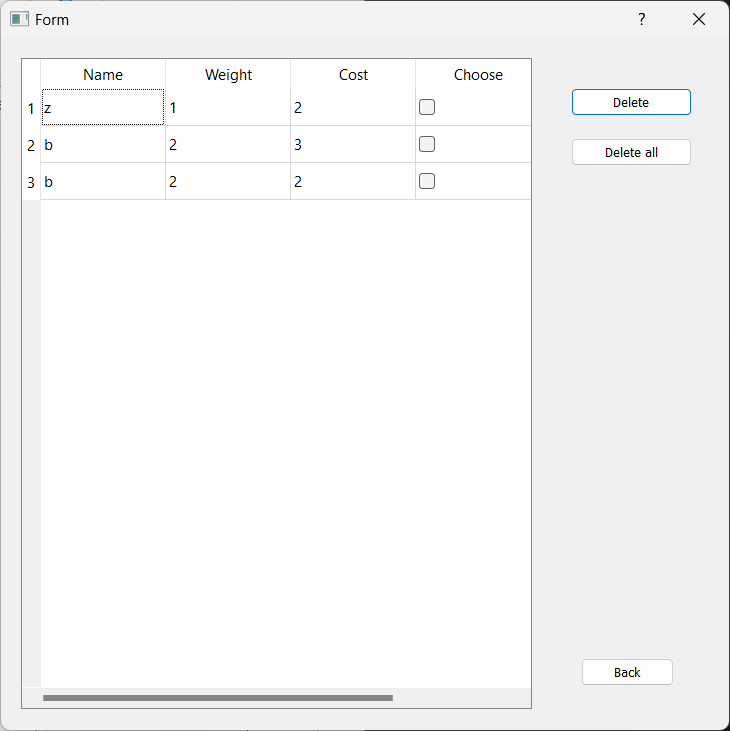
Результат:

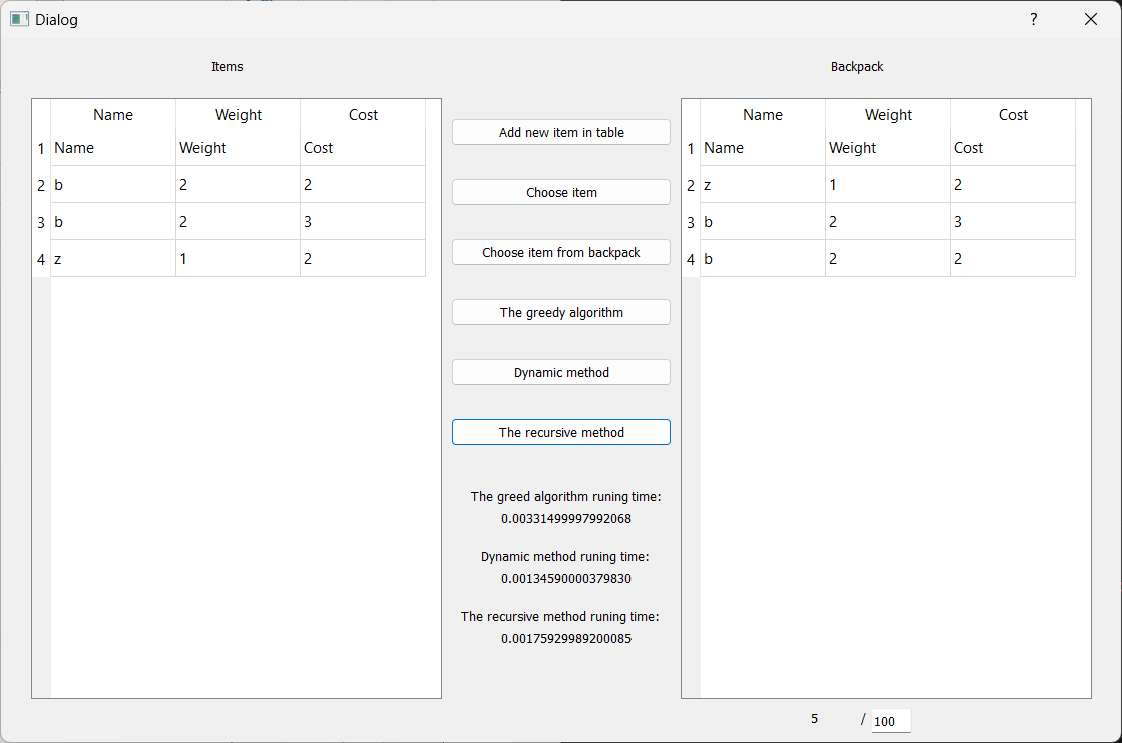
Меню:











Вывод:

В ходе данной лабораторной работы была решена задача оптимального заполнения рюкзака при помощи различных алгоритмов. В частности, было получено понимание того, как работать с жадными алгоритмами, а также, как при помощи полученных знаний оптимизировать решения множества задач.